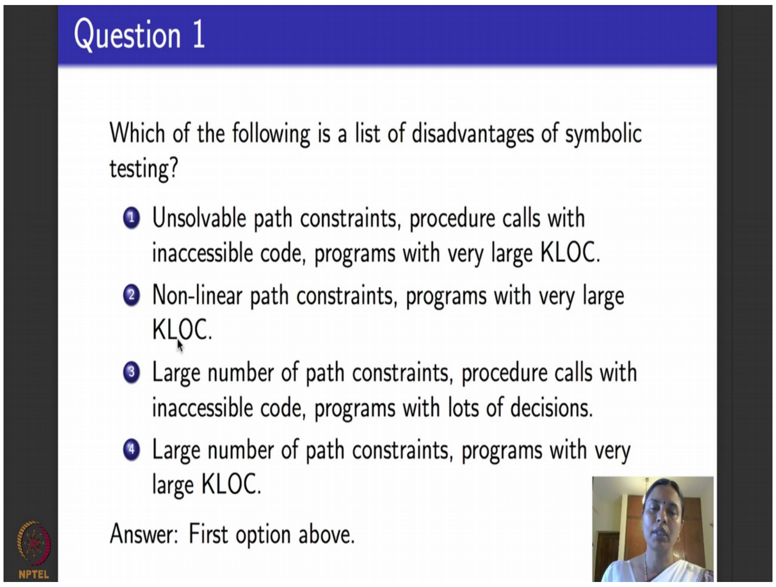
Software testing

Prof. Meenakshi D'Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore

Lecture – 59
Assignment: Week 11 Solving

Hello again, this is shall we give on assignment solving. This is the video on assignment solving for week 11, will be uploaded as always after the deadline of the assignment. If you remember week 11 lectures dealt with symbolic and concolic testing and so this assignment is on the questions related to symbolic and concolic testing.

(Refer Slide Time: 00:31)



Question 1

Which of the following is a list of disadvantages of symbolic testing?

1. Unsolvable path constraints, procedure calls with inaccessible code, programs with very large KLOC.
2. Non-linear path constraints, programs with very large KLOC.
3. Large number of path constraints, procedure calls with inaccessible code, programs with lots of decisions.
4. Large number of path constraints, programs with very large KLOC.

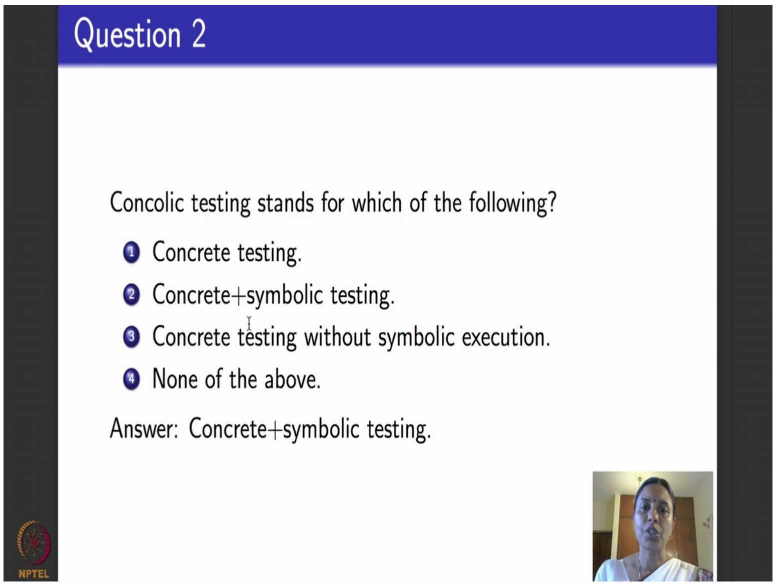
Answer: First option above.

So, as we did for other lectures I will work you through question after question explain the question and also show the answer to you. So, the first question in the video was the following made in the assignment (Refer Time: 00:44) was the following - which of the following is a list of disadvantages of symbolic testing. As I told you symbolic testing was introduced in the year 1976, but did not scale up to see the light of the day till the past decade or so, mainly because it had several disadvantages even though it was a novel technique.

So, 4 options were given to you - first one was unsolvable path constraints procedure calls with inaccessible code programs with very large lines of code, second one was non-linear path constraints programs with very large lines of code, third one was large number of path constraints procedure calls with inaccessible code programs with lots of decisions. Finally, again the same thing but two of the choices that may repeated. So, while choosing you might be confused to think that second one is a correct answer, but in options like this when there is an overlap on the choices that are available the usual correct thing is to show the complete list of disadvantages. So, from that point of view first option is the correct answer because all 3 listed here are disadvantages. Unlike option two which lists only 2 of the disadvantages and option 4 and 3 are not really disadvantages because programs with lots of decisions and the large number of path constraints practically mean the same and that just needs that a longer path constraint there should not be a disadvantage of symbolic testing.

What is a disadvantage? That if you get a path constraint that is not solvable cannot be solve by constraints solve a we cannot generate test cases and if you have procedure calls was code is not accessible then you have a problem because you do not know what additional path constraints should come from the procedure.

(Refer Slide Time: 02:31)



Question 2

Concolic testing stands for which of the following?

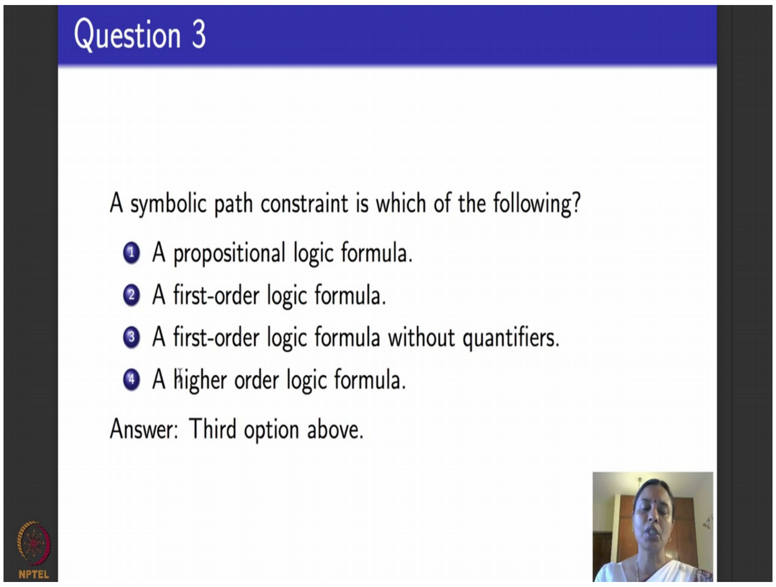
1. Concrete testing.
2. Concrete+symbolic testing.
3. Concrete testing without symbolic execution.
4. None of the above.

Answer: Concrete+symbolic testing.

So, the first one is the correct answer second question was what is concolic testing stand for. It was a simple question 4 options I given it just stands for concrete testing, concrete plus symbolic testing, concrete testing without symbolic execution or none of the above. So, the correct answer is option 2 because concolic not a first class English dictionary word, but is derived with a combination of concrete and symbolic.

Tries to do symbolic execution whenever it encounters one of the disadvantages that we discussed in question 4, it tries to substituted with concrete values hence the term concrete plus symbolic or concolic. So, the correct answers option number 2.

(Refer Slide Time: 03:21)



Question 3

A symbolic path constraint is which of the following?

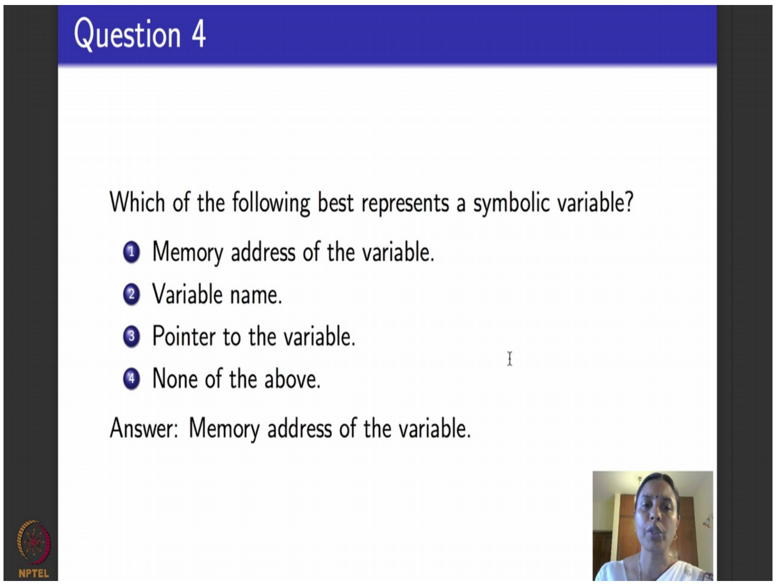
1. A propositional logic formula.
2. A first-order logic formula.
3. A first-order logic formula without quantifiers.
4. A higher order logic formula.

Answer: Third option above.

So, the third question is symbolic path constraint is which of the following, is it a propositional logic formula, is it an arbitrary first order logic formula, is it a first order logic formula that does not have any quantifiers, quantifiers are for all in their exists or is it a higher order logic formula. By higher orders I mean you could quantify over functions and relations also apart from quantifying over variables.

The correct answer is third option. Why is it a third option? It is definitely a first order logic formula not a propositional logic formula and not a higher order logic formula its first order because you have predicates of functions of several variables at the program is working on program has such functions such a first order logic formula. In addition to that there is no scope for getting paths quantifiers in the path constraint you cannot put there exists and for all. So, it is a first order logic formula without quantifiers that is the correct answer third option.

(Refer Slide Time: 04:22)


Question 4

Which of the following best represents a symbolic variable?

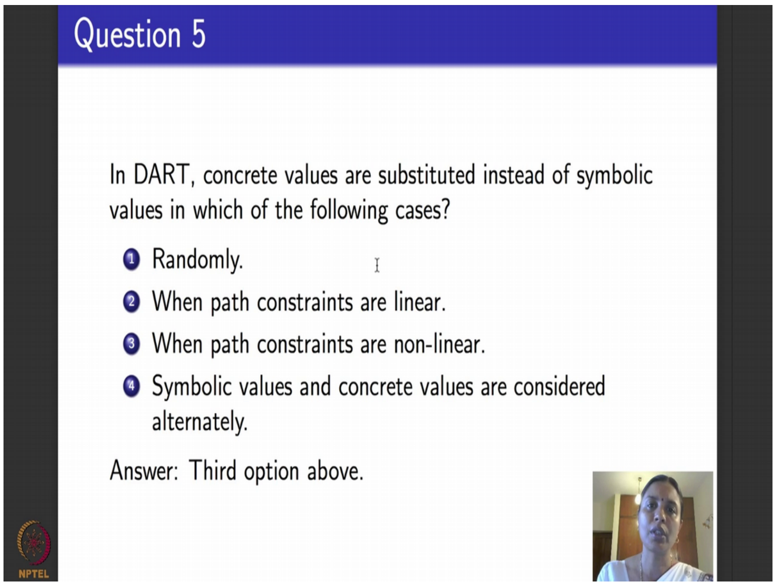
1. Memory address of the variable.
2. Variable name.
3. Pointer to the variable.
4. None of the above.

Answer: Memory address of the variable.

Fourth question says which of the following best represents a symbolic variable. We remember what a symbol variable in symbolic execution was. So, I want to be able to execute the program not with concrete values, but symbolically. Symbolic variables help me to execute a program symbolically and I my two or throughout my lectures which is gave them some variable names we called them alpha and beta or capital X and capital Y.

But if you have to actually go ahead and implement a symbolic execution tool like dart or crest or q how would you represent a symbolic variable that is the question about. So, they were 4 options again given here, it is a memory address of a variable it is just a variable name it is a pointer to a variable the fourth answer is none of the above. The correct option is the first one memory address of a variable. So, memory address of a variable can be thought of as a placeholder value that represents the variable symbolically which can be substituted by a concrete test value that satisfies the constraints that is the symbolic value satisfies, the correct answer is memory address of a variable.

(Refer Slide Time: 05:31)



Question 5

In DART, concrete values are substituted instead of symbolic values in which of the following cases?

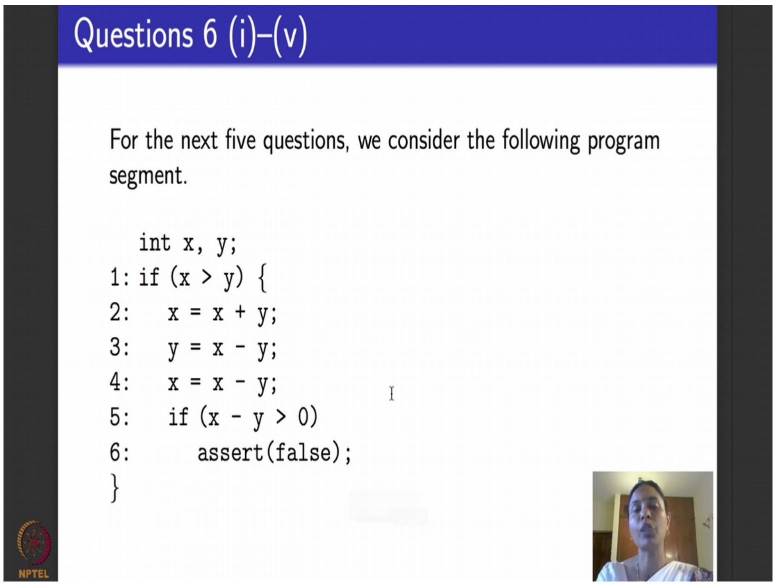
1. Randomly.
2. When path constraints are linear.
3. When path constraints are non-linear.
4. Symbolic values and concrete values are considered alternately.

Answer: Third option above.

Question number 5, in the tool dart that we saw which we discussed from the dart paper the following questions about that it ask if the concrete values are substituted instead of symbolic values in which of the following cases. They substituted randomly, concrete values are substituted when path constraints a linear, concrete values a substituted with path constraints are non-linear or the fourth option symbolic in concrete values a considered in every alternate rung. Which is the correct answer? The correct answer is the third option which is concrete values are substituted by the dart tool when path constrains are non-linear. Why is this a correct option? If you remember the time in which a paper like dart and tool was developed was in 2004-2005, at that time when it comes to constraint solving they were no good constraint solvers that could handle non-linear path constraints.

As I told you during the lectures and that week the generic problem is an undesirable problem. So, in dart they took a decision that whenever path constraints are non-linear we could substituted with concrete values and go ahead and drops symbolic execution and such cases. So, the correct answer is non-linear path constraints or option 3.

(Refer Slide Time: 06:51)



Questions 6 (i)–(v)

For the next five questions, we consider the following program segment.

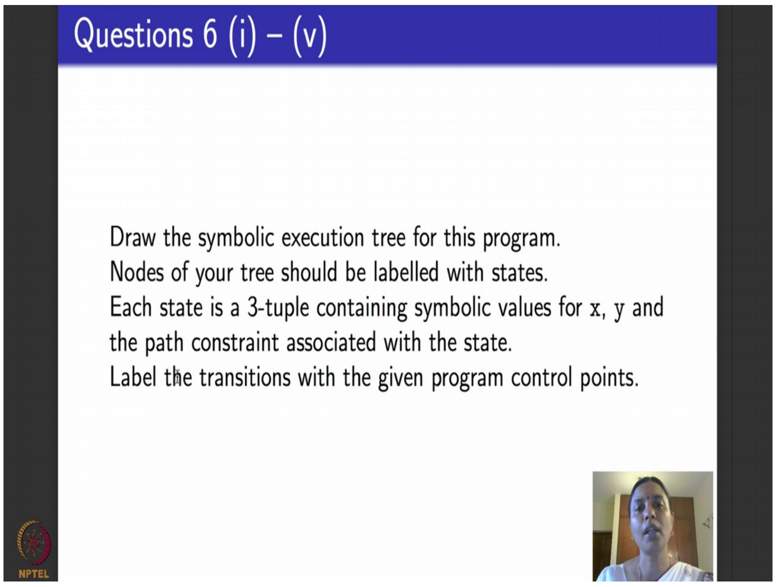
```
   int x, y;
1: if (x > y) {
2:    x = x + y;
3:    y = x - y;
4:    x = x - y;
5:    if (x - y > 0)
6:       assert(false);
   }
```

So, for the remaining 5 questions in the assignment for last week I had given you a piece of code ask you to draw the symbolic execution for that piece of code and there were questions about symbolically executing that code. Like we saw throughout this course this is not a complete piece of code is just the fragment of the code we will first go through the code that was given in the lecture and see what it is doing.

So, it took two integer variables x and y there were as an a statement which said x is greater than y then you do these 3 statements. I will come back in a minute and tell you what these 3 statements are. And then it says if x minus y is greater than zero then you assert false. So, there are two if statements one nested inside the other and then if we reaching line number 6 means that you have reached an error, assert false means the time telling that the program has an error interpreted that way; that you are asserting something that is wrong if you reach this line number 6 which means the program has an error. So, what do these lines 2, 3 and 4 the program statements to? So, what it says is if x is greater than y, let us say x is 5 y is 2 just is an example the first one does x is x plus y, it adds it. So, make x is 7 as per our values and then it does y is x minus y.

So, what is x now after the previous statement? It is x plus y. So, you substitute it here you will get x plus y minus y the plus y and minus y will get canceled and y will take the value of x. The last, the fourth line says x is x minus y. So, what is x is x minus y mean this x here on the right hand side is actually x plus y and the y is the actually x. So, what it will do is x is x plus y minus x the x and minus x will again get canceled x will become y, so basically lines 2 3 and 4 swap the values of x and y. I hope that is clear. It is not the most obvious we have swapping the most obvious we have swapping that we know is to have a temporary variable assign one value to the temporary and then do this, this just does swapping in a slightly different way that does not matter to us, but it basically swaps x and y that is all it does. And say shift swapped values of x and y are such that x minus y is greater than 0 then for some reason I have hidden error.

(Refer Slide Time: 09:25)



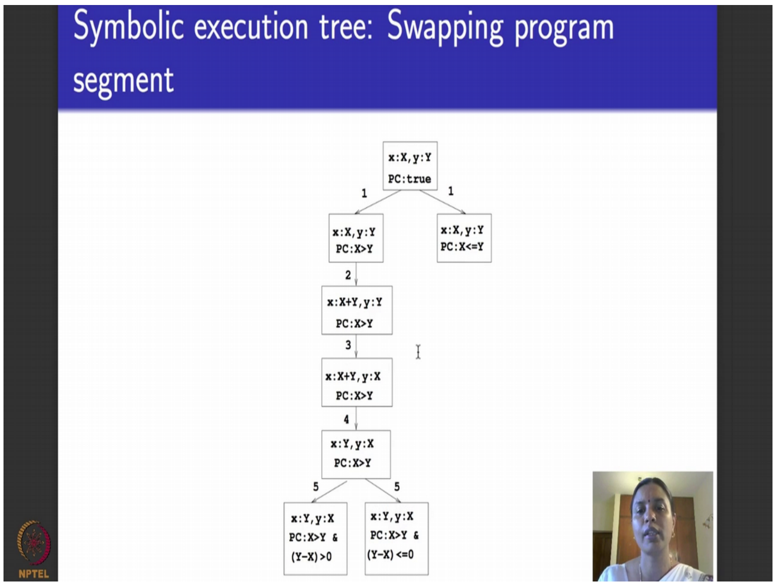
Questions 6 (i) – (v)

Draw the symbolic execution tree for this program.
Nodes of your tree should be labelled with states.
Each state is a 3-tuple containing symbolic values for x, y and the path constraint associated with the state.
Label the transitions with the given program control points.

So, this was the example of program fragment that was given and then you were asked to draw the symbolic execution tree for this program based on the symbolic execution tree there were 5 more questions asked in the assignment. So, here were some guidelines on drawing the symbolic execution tree. Symbolic execution tree is like any other graph it will have vertices and edges. So, the vertices that the nodes have here trees should be labelled with the states. What do we mean by states? If you remember what is the state and symbolic execution it is the values of the variables. So, here they were two variables x and y. So, each node of vertex should be labelled with those values and then here it explains what the status it is a value it is a 3 tuple consisting of symbolic values for x and y along with in symbolic execution we keep a path constraint which is what are all the conditions to be satisfied on the variables for me to reach a particular statement so far.

So, I keep these things in a node and the transitions of the symbolic execution tree the guideline was to label the transition with the given program control points. What do we mean by control points? They are just the statement numbers in the program. So, using these guidelines and from the lectures here is how you would have drawn a symbolic execution tree. Please do not worry, exactly you did not draw it is this way, but you would should have got something with 8 nodes that has this kind of a branching structure that would have been a correct symbolic execution tree.

(Refer Slide Time: 10:37)



Symbolic execution tree: Swapping program segment

So, what is the guideline? Let us go back and look at it the nodes the vertices of your tree should be labelled with states, state is a 3 tuple consisting of symbolic values for x and y and the path constraint.

So, if you see that is what I have done here they were two variables small x and small y, I have given their symbolic values with the names capital X and capital Y. So, here is the first state x is the variable small x is assign the symbolic value capital X the variable y small y is assigned the symbolic value capital Y and get to begin execution in the program, so as always. PC stands for path constraint to start with is true. So, if you go back and look at the code this is where I am. Very first statement in the program is a decision statement x greater than y then you do this, if x is less than or equal to y then you exit. So, there is a branch coming out of this which is labelled by the one number 1, which is the first if statement this is when x is greater than y the path constraint is added to this true and this will just be this. So, I have retained it has x greater than y. This is negation of x greater than y which is x less than or equal to y.

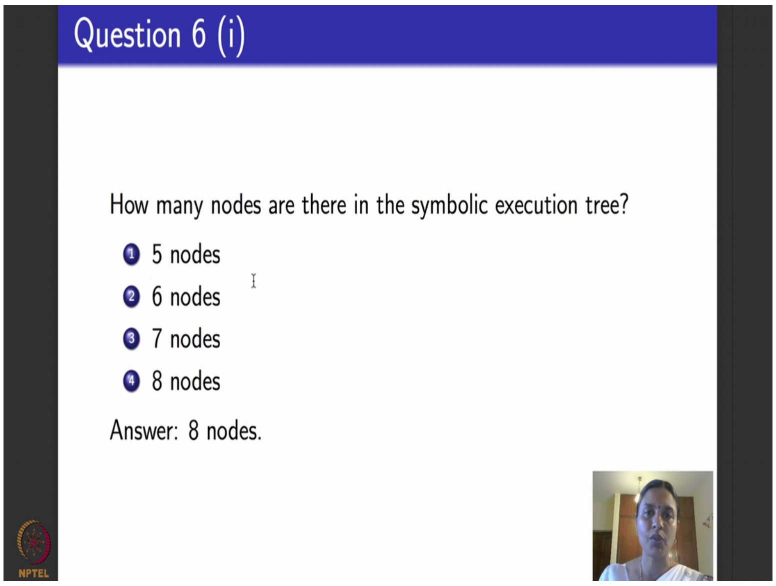
So, in the beginning first statement can test positive in which case I reach the left branch, first if statement can test negative in which case I reached the right branch. If I reach the right branch the execution stops there, there is nothing else in the program to execute. So, we go back and see what happens in the left branch when the if statement is true. So, I enter the if statement if you go back and look at the program they were 3 statements label with numbers 2 3 and 4 which we discussed they basically swapped the values of x and y. So, I am capturing them in the symbolic statement, symbolic execution tree. So, after statement number 2 is executed x becomes x plus y, y becomes y that that is true right because x becomes x plus y statement 2 does not change the value of y.

So, y states as capital y that is what is captured here path constraint is x is greater than y because I am within the first if statement. After the second statement second statement, statement number 3 changes the value of y, so x continues to stay as capital X plus capital Y, Y becomes capital X. So, this is where y gets the value of x. Path constraint is the same after the third statement you will realize now here see x is taken the value for y and y has taken the value of x. Before these 3 statements were executed you go back and look up in the node here x was taking the value x, y was taking the value y. After executing these 3 statements I have swap the values of x and y, path constraint continues to remain the same. So, where are we in the program we finished executing line number 4, statement number 4.

The statement number 5 is an if statement it says if x is greater, x minus y is greater than 0 then you assert false otherwise you exit. So, that is the symbolic execution here. So, statement 5 is an if statement. So, there is a branching in the execution tree. So, path constraint is whatever you had earlier we had x greater than y earlier if you see throughout I am ANDing that with the new path constraint which is y minus x. Why I have a changed to y minus x when x minus y was given here? Please remember after the 3 statements, 2 3 and 4 the values of x and y as walked. So, x becomes capital Y, y becomes capital X, capital X and capital Y with a symbolic execution values for small x and small y. So, path constraint is x greater than y what over they earlier and y minus x is greater than 0. I have passed the second if statement here the path constraint is x greater than y and y minus x is less than or equal to 0. So, if I pass this I reach an error I have not depicted that an my symbolic execution tree if you want to you put another node here with says that you go to line number 6 and say assert false. Because it is an error statement and asserting false reads and an abortion of execution program execution I have not depicted that.

So, this is how the symbolic execution tree will look like. So, the questions the remaining 5 questions were around this execution tree and what happens during symbolic execution.

(Refer Slide Time: 15:22)



Question 6 (i)

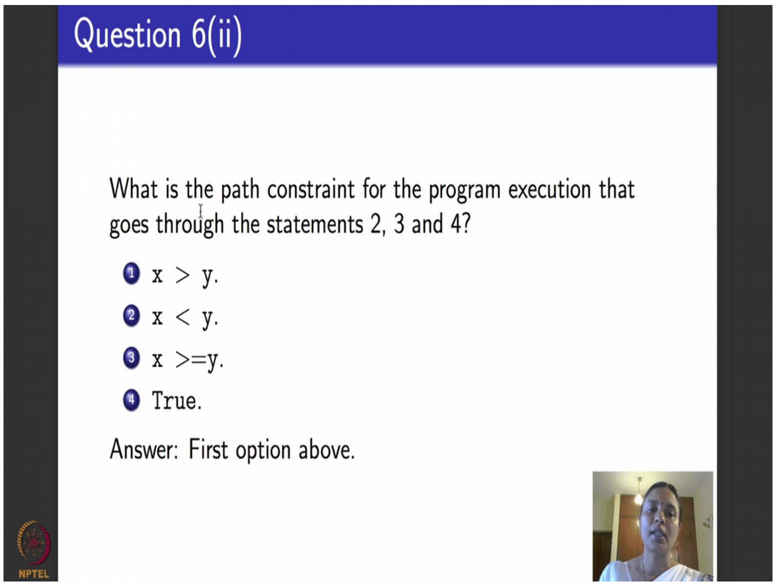
How many nodes are there in the symbolic execution tree?

1. 5 nodes
2. 6 nodes
3. 7 nodes
4. 8 nodes

Answer: 8 nodes.

So, first question with very easy if you are drawn the symbolic execution tree correctly the question just ask you how many nodes of vertices are there in the symbolic execution tree. As you could see here if you count there are 3 here, 3 plus 3 - 6 plus 2 - 8 nodes in the symbolic execution tree. So, the correct option was fourth option.

(Refer Slide Time: 15:42)



Question 6(ii)

What is the path constraint for the program execution that goes through the statements 2, 3 and 4?

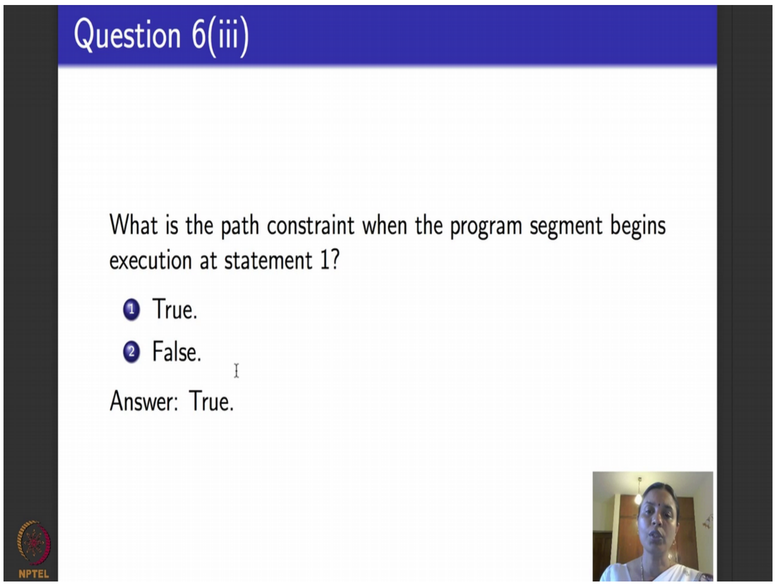
1. $x > y$.
2. $x < y$.
3. $x >= y$.
4. True.

Answer: First option above.

The next question says what is the path constraint for program execution that goes through the statements 2 3 and 4. So, let us go back statements 2 3 and 4. What was the path constraint when do I reach statements 2 3 and 4? If the if statement x greater than y turns out be true. So, the path constraint for all the 3 execution as labelled in the symbolic execution tree is x great than y.

So, there were 4 options given here x greater than y, x less than y, x greater than or equal to y or true the correct answer is x greater than y it is the first option.

(Refer Slide Time: 16:15)



Question 6(iii)

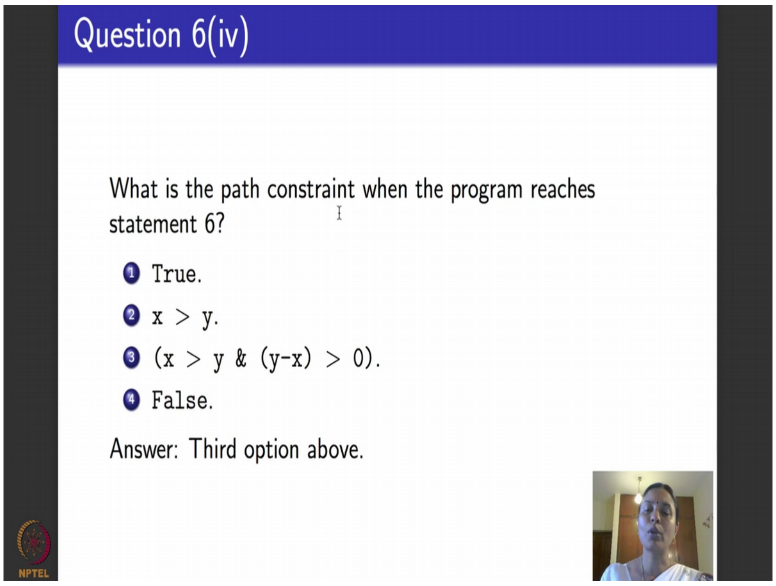
What is the path constraint when the program segment begins execution at statement 1?

1. True.
2. False.

Answer: True.

So, the third question was what was the path constraint when the program begins execution that statement number 1, here the program segment if you see there was nothing to begin with no constraint. So, the path execution when I begin execution is just the constant true that is what is given here. So, the answer is true.

(Refer Slide Time: 16:33)



Question 6(iv)

What is the path constraint when the program reaches statement 6?

1. True.
2. x > y.
3. (x > y & (y-x) > 0).
4. False.

Answer: Third option above.

The next question says what is the path constraint when the program reaches statement 6. So, 4 options were given true x greater than y, x greater than y and y minus x is greater than 0 and false the correct answer if you go back and look at the execution tree this after this is when statement number 6 is reached. So, the path constraint is x greater than y and y minus x is greater than 0 that is what is given here. So, the correct answer that you should have chosen was the third option.

This is before the second if statement this rent of write and the beginning path constraint was not false we were only asserting false. Last question was very simple, it asked will the statement number 6 in the program ever be reachable. The answer is yes it will be it is simple you do not even have to draw the symbolic execution tree we just go back and see this program, to reach statement number 6 I have to make this if statement true and I have to make this if statement true I can obviously, give values x and y such that this is true. So, statement number 6 will; obviously, be reached by the program. So, the correct answer to the last question and the assignment was yes which is the first option.

So, I hope this video helped to you to solve the assignments. Please feel free to ping me and the forum if you have any further doubts about this particular assignment.

Thank you.